TP1 Solar System - Tutoriel - Mise en place

0. Pré-requis

Installation de UnityHub et la dernière version de Unity

Création du projet

- Ouvrir UnityHub
- "New Project"
- "3D Core"
- "Project name" => "TP1 Solar System"

- "Create Project"

- Note : la fenêtre principale contient deux onglets : *"Scene"* et *"Game"*.

    - La première permet de vérifier les propriétés des objets. La seconde est une vue de la caméra au démarrage de l'application.
    - Attention à bien vérifier que *"Scene"* est sélectionné pour mettre en place la scène, sinon on a l'impression que l'interaction est bloquée. Un moyen de s'en assurer est de sauvegarder la scène : si le mode *"Play"* ou *"Pause"* a été activé, un message d'erreur apparaît.
    - Remarque : pour arrêter le mode *"Play"* ou *"Pause"*, il faut cliquer sur les icônes correspondantes pour qu'elles repassent en gris.

Project

- Dans le dossier contenant les fichiers sources :
    - identifier le dossier *"Assets"*
    - copier le dossier *"Images"* contenant des fichiers et un sous-dossier *"Materials"*
    - Vérifier que dans l'onglet "*Project*", ces dossiers apparaissent

Hiérarchie - Objets 3D

Directional Light

Supprimer cet objet de la hiérarchie. Ce sera le soleil qui fera office de source de lumière.

"Main Camera"

Une caméra est automatiquement ajoutée à la scène. On peut modifier les propriétés suivantes dans l'*"Inspector"* :

```
- Renommer en ```Main Camera```
- ```Transform```
    - ```Position : X=-1.279 ; Y = -0.02 ; Z = -1.53```
    - ```Rotation : X=0 ; Y = 30.766 ; Z = 0```

- ```Camera```
    - ```Field of View```: 27
    - ```Physical camera```: valider

- ```Add Component```
    - Sélectionner ou entrer ```Skybox```
    - Il faut créer un "*Material*" adapté :
        - Bouton droit sur *"Assets -> Images -> Materials"* et sélectionner *"Create -> Material"*
        - Donner le nom *"8k_stars_milky_way"* au *"Material"* qui apparaît (l'extension *".mat"* est automatiquement ajoutée)
        - Dans l'*"Inspector"*, les propriétés par défaut du nouveau *"Material"* sont affichées
        - Cliquer sur le petit cercle à côté de ```Albedo``` pour ouvrir le choix de textures. Dans la liste qui s'affiche, sélectionner "*8k_stars_milky_way.jpg*"
        - L'affichage de la sphère de test en bas de l'*"Inspector"* passe du blanc par défaut à la texture (noire) sélectionnéee.
    - Sélectionner de nouveau *"Main Camera"*
        - Dans le composant ```Skybox```: cliquer sur le cercle à droite du champ ```Custom Skybox``` et sélectionner *"8k_stars_milky_way"*
```

Soleil

- Dans la fenêtre *"Hierarchy"* : bouton droit -> `3D Object -> Sphere`
- Renommer en *"Soleil"*
- Dans la fenêtre *"Inspector"* :

    - ``Àdd Component**=>**Light``` (entrer le nom dans le champ de recherche
    - Modifier les propriétés de `Light`
        - `Range` : 30
        - `Mode` : Mixed => doc
        - `Intensity` : 2.6
        - `Shadow type` : Soft shadows

- Onglet *"Project"* : ouvrir le dossier *"Images -> Materials"*

    - vérifier que l'asset *"2k_sun"* apparaît
    - cliquer sur cet asset et glisser-déposer sur le soleil dans la scène : le composant `Material`se met à jour. Laisser les propriétés par défaut.
    - Noter que tous les assets sont automatiquement compilés et le résultat porte l'extension *".meta"*

- Copier le script *"JeTourne.cs"* dans le dossier *"Assets"*

    - Ajouter ce script au `Soleil` avec *"Inspector -> Add Component => Scripts => Je Tourne"*
    - Contenu actuel du fichier :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class JeTourne : MonoBehaviour
{
    // Start is called before the first frame update
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {

    }
}
```

Terre

- Ajouter un objet 3D `Sphere`dans la hiérarchie comme enfant de *"Soleil"* et le nommer "*Terre*".
- `Inspector -> Transform`
    - `Position : X = 1.5 ; Y = 0 ; Z = 0`
    - `Scale : X = 0.3 ; Y = 0.3 ; Z = 0.3` (le redimensionnement est calculé **dans** le repère local de l'objet parent (ici, le *Soleil*)
- Suivre les étapes utilisées pour affecter une texture à la Skybox plus haut pour affecter une texture à partir de l'image *"2k_earth_daymap.jpg"*
- Affecter ce `Material` directement à la `Terre` en le déplaçant depuis le `Project`sur la vue centrale
- `Inspector -> Shader`
    - `Advanced Options` : activer `Enable GPU instancing` et `Double Sided Global Illumination`

Lune

- Ajouter un objet 3D `Sphere`dans la hiérarchie comme enfant de *"Terre"* et le nommer "*Lune*".
- `Inspector -> Transform`
    - `Position : X = 0.747 ; Y = 0 ; Z = 0`
    - `Scale : X = .3 ; Y = .3 ; Z = .3`

- Reprendre la procédure d'ajout de texture à partir de l'image *"2k_moon.jpg"*. Si ce `Material` existe déjà, on peut le sélectionner directement en cliquant sur le petit rond dans `Inspector -> Mesh Renderer -> Materials`.

- Ajouter une `Camera` comme enfant de *"Lune"* et laisser les propriétés par défaut et ajouter une `Skybox` en reprenant la procédure utilisée pour la "*Main Camera*".

ATTENTION : cette seconde caméra est activée par défaut et prend la place de la première dans le `Game View`. Désactiver cette caméra en désactivant la propriété `Inspector -> Camera`.

Mars

- Ajouter un objet 3D `Sphere`dans la hiérarchie comme enfant de *"Soleil"* et le nommer "*Mars*".
- `Inspector -> Transform`
    - `Position : X = 2.27 ; Y = 0 ; Z = 0`
    - `Scale : X = 0.25 ; Y = 0.25 ; Z = 0.25`
- Reprendre la procédure d'ajout de texture à partir de l'image *"2k_mars.jpg"*. Si ce `Material` existe déjà, on peut le sélectionner directement en cliquant sur le petit rond dans `Inspector -> Mesh Renderer -> Materials`.

Deimos

- Ajouter un objet 3D `Capsule`dans la hiérarchie comme enfant de *"Mars"* et le nommer "*Deimos*".
- `Inspector -> Transform`
    - `Position : X = .92 ; Y = 0 ; Z = 0`
    - `Scale : X = 0.169 ; Y = 0.169 ; Z = 0.169`
- Reprendre la procédure d'ajout de texture à partir de l'image *"deimoscyl4.jpg"* site. Si ce `Material` existe déjà, on peut le sélectionner directement en cliquant sur le petit rond dans `Inspector -> Mesh Renderer -> Materials`.

Phobos

- On veut ajouter un objet 3D depuis une source extérieure. Quelle que soit cette source, il faut transformer cet objet en *"Unity asset"*
- Le plus rapide est de vérifier sur (l'Asset Store)[https://assetstore.unity.com/] si cet objet s'y trouve déjà
- Sinon, on peut récupérer un objet dans un format 3D quelconque et utiliser un *Package* pour l'importer
- Dans cet exemple, le fichier est récupéré sur (ce site)[https://solarsystem.nasa.gov/resources/2358/phobos-3d-model/] et défini au format (glTF)[https://en.wikipedia.org/wiki/GlTF].
- On commence par utiliser le `Window -> Packet Manager` pour ajouter une librairie permettant d'importer ce format.
- La librairie ("*GLTFUtility*")[https://github.com/Siccity/GLTFUtility] est retenue ici
    - Ouvrir le `Packet Manager`
    - Cliquer sur l'icône "+" d'ajout et entrer l'URL `https://github.com/siccity/gltfutility.git`
    - *Unity* charge cette librairie et affiche des messages de type "*Experimental Packages in Use*" => ne semble pas gênant pour la suite
    - Lorsque ce package a été installé, placer l'objet représentant *Phobos* dans le dossier "*Assets -> Images*"
    - *Unity* lit correctement ce fichier (en fait, une archive constituée de plusieurs éléments) et le transforme en *Asset*.
    - Glisser-déposer ce fichier dans la hiérarchie, comme enfant de *Mars*
- `Inspector -> Transform`
    - `Position : X = -.7 ; Y = 0 ; Z = 0`
    - `Scale : X = 0.01 ; Y = 0.01 ; Z = 0.01`

Repositionner la caméra

Modifier les paramètres de rotation et de la lingueur focale de la caméra pour visualier le soleil et plusieurs planètes.

Hiérarchie - Canvas

L'objectif est maintenant d'ajouter une interface 2D en superposant une interface graphique à la scène.

- Dans la zone "*Hierarchy*", bouton droit : `UI -> Canvas`. Un objet `EventSystem` est également automatiquement ajouté.
- Parmi les propriétés de ce "*Canvas*", noter que les valeurs de position et de dimensions sont automatiquement définies et non modifiables. Ces valeurs seront adaptées au contenu de cet objet.
-
    - `Inspector -> Rect Transform`
    - `Pos X = 459 ; Pos Y = 257 ; Pos Z = 0`
    - `Width = 918 ; Height = 514`

PanelSoleil

- Ajouter un `UI -> Panel` nommé *"PanelSoleil"* commen enfant de *"Canvas"*.
- `Inspector -> Rect Transform`
    - cliquer sur l'icone de positionnement en haut à gauche, marquée `Stretch / Stretch` et sélectionner `top / left`
    - `Pos X = 180 ; Pos Y = -80 ; Pos Z = 0`
    - `Width = 300 ; Height = 130`
    - `Pivot X = 0 ; Y = 1`
    - => on peut modifier ces valeurs, du moment que ce *Panel* se retrouve bien à l'intérieur de *"Canvas"*.

Nom du panel

- Ajouter un `UI -> Legacy -> Text` nommé *"NomPanelSoleil"* comme enfant de *"PanelSoleil"*.
- `Inspector -> Rect Transform`
    - `Stretch / Stretch` -> `top / left`
    - `Pos X = 100 ; Pos Y = -20 ; Pos Z = 0`
    - `Width = 160 ; Height = 30`
    - `Pivot X = 0 ; Y = 1` (fait passer `Pos Y` à -5)
- `Inspector -> Text`
    - `Text -> Soleil`
    - `Paragraph -> Alignment -> milieu vertical`

Case à cocher

- Ajouter un `UI -> Toggle` nommé *"TogglePanelSoleil"* comme enfant de *"PanelSoleil"*.
- `Inspector -> Rect Transform`
    - `Stretch / Stretch` -> `top / left`
    - `Pos X = 100 ; Pos Y = -45 ; Pos Z = 0`
    - `Width = 160 ; Height = 20`
    - `Pivot X = 0 ; Y = 1` (fait passer `Pos X` à 20 et `Pos Y` à -35)

"TogglePanelSoleil" contient lui-même plusieurs enfants : - `Background --> Checkmark - Label` : renommer "*LabelTogglePanelSoleil*" et modifier la propriété `Text` pour remplacer "*Toggle*" par "*Animer*"

Label Vitesse

- Ajouter un `UI -> Legacy -> Text` nommé *"VitesseAnimSoleil"* comme enfant de *"PanelSoleil"*.
- Modifier les propriétés de `Inspector -> Rect Transform` pour aligner ce label sous les précédents (modifier d'abord la valeur du pivot avant les coordonnées de cet objet)
- `Inspector -> Text`
    - `Text -> Vitesse`
    - `Paragraph -> Alignment -> milieu vertical`

Slider Vitesse

- Ajouter un `UI -> Slider` nommé *"SliderVitesseAnimSoleil"* comme enfant de *"PanelSoleil"*.
- Placer cet objet en-dessous des autres, collé à *"VitesseAnimSoleil"*
- Modifier sa largeur pour la faire passer à 250
- Ne pas modifier les propriétés de ses enfants : `Background`, `Fill Area` et `Handle Slide Area`
- Laisser la propriété `Min Value` à 0 et passer la propriété `Max Value` à 100.

Panel Terre

- Dupliquer "*PanelSoleil*" avec `UI -> Duplicate` dans la hiérarchie en deux exemplaires : "*PanelTerre*" et "*PanelMars*"
- Vérifier ques les propriétés des objets dupliqués sont bien mises à jour (par exemple pour les valeurs min et max des sliders)
- Déplacer ces panels les uns en dessous des autres dans le *Canvas* principal
- Renommer les objets en conséquence

Animation

JeTourne pour Soleil

- Modifier le script *JeTourne.cs* de la manière suivante et l'affecter à *Soleil* :

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class JeTourne : MonoBehaviour
{

    public bool mustTurn = true;

    // Start is called before the first frame update
    void Start()
    {
        Debug.Log("Start() de " + gameObject.name);
        mustTurn = true;
    }

    // Update is called once per frame
    void Update()
    {

    }

    private void FixedUpdate()
    {
```

```
        Debug.Log("FixedUpdate-) de " + gameObject.name);
        if (mustTurn)
        {
            Debug.Log("Rotation de " + gameObject.name);
            transform.Rotate(0, 5 * Time.fixedDeltaTime, 0);
        }
    }


    public void toggle()
    {
        Debug.Log("toggle() de " + gameObject.name);
        mustTurn = !mustTurn;
    }

}
```

- Noter que l'attribut `Must Turn` apparaît automatiquement sous forme de "toggle" dans l'`Inspector`.
- Ajouter le même script à chacune des planètes :
    - `Inspector -> Add Component`
    - Dans le champ texte, entrer le nom du Script

JeTourne pour les autres planètes

- Affecter le même script "*JeTourne.cs*" aux autres planètes
- Démarrer le `GameView` et désactiver à tour de rôle le *toggle* `Must Turn` de chaque *GameObject* pour mieux observer ses effets.

Scripts associés aux Canvas

PanelSoleilScript.cs

- Associer le script suivant au `GameObject` *PanelSoleil*. Il permet de modifier via ce `Canvas` l'activation/désactivation de la rotation du soleil et sa vitesse.

```
// Script attaché aux GameObject "PanelSoleil" pour gérer l'animation du soleil


// Inspiré de
// https://docs.unity3d.com/2019.1/Documentation/ScriptReference/UI.Toggle-onValueChanged.html
//Attach this script to a Toggle GameObject. To do this, go to Create>UI>Toggle.
//Set your own Text in the Inspector window

using UnityEngine;
using UnityEngine.UI;

public class PanelSoleilScript : MonoBehaviour
{
    Toggle m_Toggle;
    Slider m_Slider;
    GameObject soleil;

    void Start()
    {
        // m_Toggle est le Toggle enfant de PanelSoleil
       m_Toggle = GetComponentInChildren<Toggle>();
        if (m_Toggle == null)
            Debug.Log("m_Toggle Soleil = nul");

        // m_Slider est le Slider enfant de PanelSoleil
       m_Slider = GetComponentInChildren<Slider>();
        if (m_Slider == null)
            Debug.Log("m_Slider soleil = nul");

        //Add listener for when the state of the Toggle changes, to take action
        m_Toggle.onValueChanged.AddListener(delegate {
            ToggleValueChanged(m_Toggle);
        });

        m_Slider.onValueChanged.AddListener(delegate {
            SliderValueChanged(m_Slider);
        });

        soleil = GameObject.Find("Soleil");
    }

    void ToggleValueChanged(Toggle change)
    {
      soleil.GetComponent<JeTourne>().mustTurn = !soleil.GetComponent<JeTourne>().mustTurn;
    }

    void SliderValueChanged(Slider change)
    {
      soleil.GetComponent<JeTourne>().rotationSpeed = change.value;
    }
}
```

Adaptation du script à Terre et Mars

- Reprendre le script ci-dessus et modifier les occurrences de "*Soleil*" par "*Terre*" ou "*Mars*".
- Une fois ces scripts compilés, les ajouter respectivement à "*PanelTerre*" et "*PanelMars*" en passant par l'`Inspector -> Add Component -> Scripts` (les scripts doivent apparaître dans la liste).
- Ne pas oublier de passer la `Max Value` de `SliderVitesseAnimTerre` et `SliderVitesseAnimMars` à 100.

Manipulation des caméras

Ajout d'un Panel

- Lorsque plusieurs caméras sont utilisées, la caméra visible est celle dont la valeur de la propriété `depth` est la plus élevée, d'après la (doc)[https://docs.unity3d.com/Manual/MultipleCameras.html].
- Si ce n'est déjà fait, ajouter un `GameObject Camera` "*CameraLune*" à la *Lune*.
- Dans le *Canvas* principal, ajouter un `Panel` nommé "*PanelCameras*" qui comprendra un `Toggle` nommé "*ToggleCameras*", dont le `Label` contiendra le texte "*Main Camera / Moon Camera*".
- Aligner "*PanelCameras*" avec les autres `Panels` de *Canvas*.

Script de permutation entre les caméras

Associer le script suivant (nommé "*PermutationCamerasScript*") à *PanelCameras*.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

// Inspiré de https://docs.unity3d.com/Manual/MultipleCameras.html

public class PermutationCamerasScript : MonoBehaviour
{
    // Toggle du Panel associé à ce script
    Toggle m_Toggle;

    // Caméras manipulées
    Camera mainCamera;
    Camera moonCamera;

    // Start is called before the first frame update
    void Start()
    {
        // m_Toggle est le Toggle enfant de PabelCameras
       m_Toggle = GetComponentInChildren<Toggle>();
        if (m_Toggle == null)
            Debug.Log("m_Toggle PanelCameras = nul");
```

```
        //Add listener for when the state of the Toggle changes, to take action
        m_Toggle.onValueChanged.AddListener(delegate {
            ToggleValueChanged(m_Toggle);
        });

        mainCamera = getCameraFromParentName("MainCamera");
        mainCamera.enabled = true;

        moonCamera = getCameraFromParentName("CameraLune");
        moonCamera.enabled = false;
    }

    private Camera getCameraFromParentName(System.String parentName) {
        GameObject cameraObject = GameObject.Find(parentName);
        if (cameraObject == null)
            Debug.Log("CameraObject of " + parentName +  "= nul");

        Camera camera = cameraObject.GetComponent<Camera>();
        if (camera == null)
            Debug.Log("Camera of " + parentName +  "= nul");

        return camera;
    }

    void ToggleValueChanged(Toggle change)
    {
      mainCamera.enabled = !mainCamera.enabled;
      moonCamera.enabled = !moonCamera.enabled;
    }

}
```

3ème caméra "vue du haut"

- Dans la hiérarchie, ajouter une `Camera` et la nommer "*CameraUpView*"
- La placer et l'orienter de manière à offrir une "vue de dessus" du système solaire :
  - `Transform -> Position -> X = 0.36 ; Y = 12.01 ; Z = -0.58`
  - `Transform -> Orientation -> X = 84.43 ; Y = -20.639 ; Z = -124.044`
  - `Camera -> Field of View = 28`
  - `Depth = 1` => valeur supérieure à la propriété de même nom de "*MainCamera*" afin de placer cette vue au-dessus de l'autre
  - `ViewPort Rect -> X = 0.8 ; Y = 0` => place le rectangle de cette vue en bas à droite de la scène finale
  - `ViewPort Rect -> W = 0.2 ; H = 0.2` => facteurs d'échelle des dimensions du rectangle de vue
- Ajouter un `Component SkyBox` utilisant *"8k_stars_milky_way"*

Trajectoires

L'objectif est de créer un objet qui tourne autour du soleil mais n'appartient pas à sa hiérarchie.

- Importer un objet 3D au format `.obj` pour représenter un nouvel astéroide
  - Placer cet objet dans le répertoire *Assets* (ou un sous-dossier) dans `Project`
  - Déplacer cet objet depuis `Project` vers `Hierarchy`, ce qui l'ajoute à la scène.
  - Nommer cet objet "*Comet*"
  - Cet objet contient un sous-objet nommé "*default*"
  - Associer comme enfant à ce "**default" un nouvel objet avec le menu du bouton droit : `Effects -> Trail`
  - Modifier la `Position` de ce `Trail` en `X = 0 ; Y = 0 ; Z = 0` pour le "coller" sur *default*
  - Modifier le `Trail Renderer` de ce `Trail`
    - en diminuant sa largeur `Width` (valeur 0.2 par exemple)
    - en modifiant éventuellement sa couleur
    - en modifiant la propriété `Time` correspondant à la durée de vie de la trace laissée par l'objet.

Le script suivant permet de créer une trajectoire elliptique autour du centre du repère global, avec le soleil constituant l'un des foyers de l'ellipse. Il est possible de modifier la position du soleil sur l'axe X durant l'animation pour observer les différents effets, mais cette position ne doit pas excéder le demi-grand axe de l'ellipse.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UIElements;

// Trajectoire circulaire ou elliptique en utilisant le soleil comme foyer d'une ellipse

// https://vanoise49.pagesperso-orange.fr/annexe_1_Trajectoire%20elliptique.pdf

public class TrajectoiresScript : MonoBehaviour
{
        private GameObject soleil;
        public float semiMajor = 20f ; // demi-grand axe de l'ellipse > 0

        // angle de rotation entre ce GameObject et le soleil ; à calculer à chaque pas de temps (
        public float angle = 0f; // à convertir en radians

        // inclinaison de l'ellipse (en degrés) en restant dans le même plan de l'écliptique
        public float eclipticAngle = 0f; // à convertir en radians

        // L'excentricité est définie selon la distance entre le centre de l'ellipse, l'un de ses foyers et le demi-grand axe
        private float eccentricity;

    // Start is called before the first frame update
    void Start()
    {
                soleil = GameObject.Find("Soleil");
                if (soleil == null) {
                        Debug.Log("TrajectoiresScript: soleil = null");
                }
    }

    // Update is called once per frame
    void Update()
    {
    }

    // FixedUpdate is called at fixed frames. To use with rigid Bodies
    // https://docs.unity3d.com/ScriptReference/MonoBehaviour.FixedUpdate.html
    void FixedUpdate()
    {
                // En toute généralité, l'excentricité est définie sur un axe, X par convention.
                // Formule générale : excentricité = Abs(centre de l'ellipse.X - foyer de l'ellipse.X) / demi-grand axe
                // => et cette valeur doit être comprise dans [0 ; 1[
                // On suppose ici que le centre de l'ellipse est le centre du repère global (0,0,0)
                eccentricity = Mathf.Abs(soleil.transform.position.x) / semiMajor;
                Debug.Log("eccentricity = " + eccentricity);

                //  semiMinor = demi-petit axe de l'ellipse
                //   => semiMajor et semiMinor sont liés à l'excentricité de l'ellipse E par semiMinor = semiMajor * sqrt(1 - E*E)
                //  => si semiMajor et semiMinor sont égales, l'orbite est un cercle (facile à voir avec tiltEclipticAngle = 0)
                float semiMinor = semiMajor *Mathf.Sqrt(1 - eccentricity * eccentricity);

                float angleRad = Mathf.Deg2Rad * angle;
                float angleRadCos = Mathf.Cos(angleRad);
                float angleRadSin = Mathf.Sin(angleRad);

                float eclipticAngleRad = Mathf.Deg2Rad * eclipticAngle;
                float eclipticAngleRadCos = Mathf.Cos(eclipticAngleRad);
                float eclipticAngleRadSin = Mathf.Sin(eclipticAngleRad);

        // Remarque: Multiplier par Time.fixedDeltaTime (=0.02) donne des résultats proches de 0
        // les coordonnées X et Z résultantes sont donc proches de 0 => ce GameObjet ne se déplace donc quasiment pas
        // => on "équilibre les coordonnées en multipliant par 50
                this.gameObject.transform.position = new Vector3(((semiMajor * angleRadCos * eclipticAngleRadCos) - (semiMajor * angleRadSin * eclipticAngleRadSin)) * Time.fixedDeltaTime * 50f,
                                                                 0,
                                                                 ((semiMinor * angleRadCos * eclipticAngleRadSin) - (semiMinor * angleRadSin * eclipticAngleRadCos)) * Time.fixedDeltaTime * 50f);
```

```
                angle += 1f;//can be used as speed
                if (angle > 360f) {
                        angle = 0f;
                }

                Debug.Log("new X = " + this.gameObject.transform.position.x + ", new Z = " + this.gameObject.transform.position.z);
                Debug.Log("soleil.X = " + soleil.transform.position.x); // On peut faire varier la position du Soleil en X, sans dépasser la valeur de semiMajor
                Debug.Log("Time.fixedDeltaTime = " + Time.fixedDeltaTime);
    }
}
```

Déplacements de caméras

Création de ToggleGroup

L'objectif est de pouvoir sélectionner la planète sur laquelle la vue de la caméra principale sera centrée, ou bien de revenir à l'orientation originale de la caméra.

Source : https://www.youtube.com/watch?v=6QJ789LOcu8

- Repositionner les `Panel` du `Canvas` principal pour ajouter un nouveau `Panel` nommé "*PanelLookAt*"
- Ajouter dans ce `Panel` un nouveau label nommé *NomPanelLookAt* et contenant le `Text` *Cameras LookAt*
- Ajouter une succession de `Toggle` avec les `Text` respectifs : *Soleil*, *Terre*, *Mars* et *Original*
- Pour chaque `Toggle` :
  - Désactiver la propriété `IsOn`, sauf pour *Original*
  - Ajouter le composant `Layout Element` et spécifier la propriété `LayoutElement -> Preferred Height` à 20 (coche activiée)
  -

On va maintenant regrouper tous ces `Toggle` dans un `ToggleGroup`.

- Dans *PanelLookAt*, créer un `Create Empty` que l'on nommera *ToggleGroupLookAt*. Il s'agit d'un `GameObject` quasiment vide avec des propriétés minimales, que l'on va utiliser comme un conteneur.
- Modifier la hiérarchie pour que *ToggleGroup* devienne le parent des `Toggle`.
- Ajouter le composant `Vertical Layout Group` à *ToggleGroupLookAt*
  - Dans ce composant, désactiver la propriété `Child Force Expand -> Height`
- Ajouter le script *ToggleGroupLookAt* à `Toggle Group`

```
// https://www.youtube.com/watch?v=0b6KmdPcDQU

// Associer ce script au ToggleGroup dédié au centrage de la vue de la caméra sur une planète donnée

using UnityEngine;
using System.Collections;
using UnityEngine.UI;
using System.Linq;

public class ToggleGroupLookAtScript : MonoBehaviour {
        ToggleGroup m_ToggleGroup;
        Toggle m_PreviousToggle;
        Toggle m_SelectedToggle;

        public Toggle GetCurrentSelection() {
                return m_ToggleGroup.ActiveToggles().FirstOrDefault();
        }

        public Toggle GetPreviousSelection() {
                return m_PreviousToggle;
        }

        void Start() {
                m_ToggleGroup = GetComponent<ToggleGroup>();
                m_PreviousToggle = GetCurrentSelection();
                m_SelectedToggle = GetCurrentSelection();
                Debug.Log("[ToggleGroupLookAtScript][Start] m_PreviousToggle = " + m_PreviousToggle.name + ", m_SelectedToggle = " + m_SelectedToggle.name);

        }

        public  GameObject GetGameObjectFromToggle(Toggle toggle) {
                switch(toggle.name) {
                        case "ToggleSoleil":
                                return GameObject.Find("Soleil");
                        case "ToggleTerre":
                                return GameObject.Find("Terre");
                        case "ToggleMars":
                                return GameObject.Find("Mars");
                        case "ToggleComet":
                                return GameObject.Find("Comet");
                        case "ToggleOriginal":
                                return GameObject.Find("MainCamera");
                        default:
                                return null;
                }
        }


        void Update() {
                if (GetCurrentSelection() != m_SelectedToggle) {
                        m_PreviousToggle = m_SelectedToggle;
                        m_SelectedToggle = GetCurrentSelection();
                        Debug.Log("[ToggleGroupLookAtScript][Update] m_PreviousToggle = " + m_PreviousToggle.name + ", m_SelectedToggle = " + m_SelectedToggle.name);

                }
        }
}
```

- Sélectionner l'ensemble des `Toggle` enfants, puis cliquer sur *ToggleGroup* et le déplacer à la souris dans la propriété `Toggle -> Group` des enfants sélectionnés. La valeur de cette propriété passe de `None (Toggle Group)` à `ToggleGroup(Toggle GroupLookAt)`
- On peut tester l'activation des boutons en démarrant la scène et en cliquant sur chaque bouton du groupe, ce qui désactive le bouton précédent.

Utilisation de LookAt avec interpolation sphérique

- Pour chacun des `Toggle` enfants, le script "*LookAtPlanetScriptSlerp.cs*" permet de positionner en continu le centre de vue de la caméra sur la planète sélectionnée, ou revenir à l'orientaton originale.
- Remarquer que le script est le même pour tous les boutons, on détecte quelle est la planète choisir directement dans le script *ToggleGroupLookAtScript.cd*.

```
// Script attaché aux GameObject "Toggle" de PanelLookAt pour centrer la vue de la caméra principale sur une planète donnée, ou revenir à l'orientation originale. Utilisation d'une interpolation

// Attention : chaque Toggle possède sa propre instance de ce script.
// Donc toutes les variables du script sont définies indépendamment pour chaque Toggle (elles ne sont pas partagées


// Inspiré de
// https://docs.unity3d.com/2019.1/Documentation/ScriptReference/UI.Toggle-onValueChanged.html
//Set your own Text in the Inspector window

using UnityEngine;
using UnityEngine.UI;
using System.Collections;
using System.Linq;

public class LookAtPlanetScriptSlerp : MonoBehaviour
{
    Toggle m_Toggle;
        GameObject m_MainCamera;
        float m_TimeCount = 0.0f;
        bool m_EnableInterpolation = false;
        ToggleGroup m_ToggleGroup;
        Quaternion m_MainCameraInitialRotation;

        public ToggleGroupLookAtScript m_ToggleGroupLookAtScript;
```

```
    void Start()
    {
                m_ToggleGroupLookAtScript = GameObject.FindObjectOfType(typeof(ToggleGroupLookAtScript)) as ToggleGroupLookAtScript;

                // ToggleGroup parent
                m_ToggleGroup = GetComponentInParent<ToggleGroup>();
                if (m_ToggleGroup == null) {
                        Debug.Log("[LookAtPlanetScriptSlerp] m_ToggleGroup = nul");
                }
                else {
                        Debug.Log("[LookAtPlanetScriptSlerp] m_ToggleGroup = " + m_ToggleGroup.name);
                }

        // m_Toggle est le Toggle enfant de ce GameObject
       m_Toggle = GetComponentInChildren<Toggle>();
        if (m_Toggle == null)
            Debug.Log("[LookAtPlanetScriptSlerp] m_Toggle = nul");

        //Add listener for when the state of the Toggle changes, to take action
        m_Toggle.onValueChanged.AddListener(delegate {
            ToggleValueChanged(m_Toggle);
        });

                m_MainCamera = GameObject.Find("MainCamera");
                m_MainCameraInitialRotation = m_MainCamera.transform.rotation;
                if (m_MainCamera == null) {
            Debug.Log("[LookAtPlanetScriptSlerp] m_MainCamera = nul");
                }

    }

    void ToggleValueChanged(Toggle change)
    {
                if (m_Toggle.isOn) {
                        Debug.Log("[LookAtPlanetScriptSlerp] m_Toggle = " + m_Toggle.isOn + " pour " + name);
                        m_EnableInterpolation = true;
                }
    }

        // G
        void FixedUpdate() {
                Toggle selectedToggle = m_ToggleGroupLookAtScript.GetCurrentSelection();
                Toggle previousToggle = m_ToggleGroupLookAtScript.GetPreviousSelection();

                if (m_Toggle.isOn) {
                        GameObject selectedObject = m_ToggleGroupLookAtScript.GetGameObjectFromToggle(selectedToggle);
                        if (m_EnableInterpolation) {
                                if (selectedToggle != previousToggle) {

                                        if (selectedObject != m_MainCamera) {
                                                GameObject previousObject = m_ToggleGroupLookAtScript.GetGameObjectFromToggle(previousToggle);
                                                if (selectedObject == null || previousObject == null) {
                                                        Debug.Log("[LookAtPlanetScriptSlerp][Update] selectedObject = nul ou previousObject = nul");
                                                }

                                                m_MainCamera.transform.rotation =
                                                        Quaternion.Slerp(m_MainCamera.transform.rotation,
                                                                                                Quaternion.LookRotation(selectedObject.transform.position -
                                                                                                                                                        m_MainCamera.transform.position),
                                                                                                m_TimeCount);
                                                //                                      Debug.Log("[LookAtPlanetScriptSlerp][Update] rotation = " + m_MainCamera.transform.rotation);
                                                m_TimeCount += Time.deltaTime;
                                                // Debug.Log("[LookAtPlanetScriptSlerp][Update] m_GameObject = " + m_GameObject);
                                                if (m_TimeCount > 1.0f) {
                                                        m_TimeCount = 0.0f;
                                                        m_EnableInterpolation = false;
                                                }
                                        }
                                        else { // selectedObject == m_MainCamera
                                                GameObject previousObject = m_ToggleGroupLookAtScript.GetGameObjectFromToggle(previousToggle);
                                                if (selectedObject == null || previousObject == null) {
                                                        Debug.Log("[LookAtPlanetScriptSlerp][Update] selectedObject = nul ou previousObject = nul");
                                                }

                                                m_MainCamera.transform.rotation = Quaternion.Slerp(m_MainCamera.transform.rotation,
                                                                                                                                                                m_MainCameraInitialRotation,
                                                                                                                                                                m_TimeCount);
                                                //                                      Debug.Log("[LookAtPlanetScriptSlerp][Update] rotation = " + m_MainCamera.transform.rotation);
                                                m_TimeCount += Time.deltaTime;
                                                // Debug.Log("[LookAtPlanetScriptSlerp][Update] m_GameObject = " + m_GameObject);
                                                if (m_TimeCount > 1.0f) {
                                                        m_TimeCount = 0.0f;
                                                        m_EnableInterpolation = false;
                                                }

                                        }
                                }
                        }
                        else if (m_MainCamera != selectedObject) {
                                m_MainCamera.transform.LookAt(selectedObject.transform);
                        }
                }
        }
}
```